Modeling and Generating Moving Trees from Video

Chuan Li[†] Oliver Deussen[‡] Yi-Zhe Song[†] Phil Willis[⋆] Peter Hall[†]

Media Technology Research Centre, University of Bath, UK[†] University of Konstanz, Germany[‡]
Centre for Digital Entertainment, University of Bath, UK[⋆]

Figure 1: *Using a single video as input (left, background removed using alpha matting), our system outputs a 3D dynamic tree model (middle). Using the model, potentially an infinite number of unique trees with similar appearance and motion can be generated (right).*

Abstract

We present a probabilistic approach for the automatic production of tree models with convincing 3D appearance and motion. The only input is a video of a moving tree that provides us an initial dynamic tree model, which is used to generate new individual trees of the same type. Our approach combines global and local constraints to construct a dynamic 3D tree model from a 2D skeleton. Our modeling takes into account factors such as the shape of branches, the overall shape of the tree, and physically plausible motion. Furthermore, we provide a generative model that creates multiple trees in 3D, given a single example model. This means that users no longer have to make each tree individually, or specify rules to make new trees. Results with different species are presented and compared to both reference input data and state of the art alternatives.

Keywords: tree modeling and animation, generative model.

Links: DL PDF

1 Introduction

Trees are among the Earth's most useful and beautiful products of nature. They have been drawn, painted and modeled for centuries. Contemporary tools make it possible to produce high quality 3D moving models. Typically though, each tree must be individually made by an expert user either by sketching or by providing suitable images — new trees can be grown automatically only if abstract rules are defined. The difficulties of building a tree are magnified when the tree is to move. Overall, tree modeling remains a time consuming process that often relies on expert knowledge.

In this paper we address the tree modeling problem using an approach that is almost entirely automatic. To make a model, the user only has to outline the tree in an initial video frame. The system then creates a full 3D model including motion. This model furthermore can serve as an example to automatically generate new 3D dynamic tree models of the same species. Our approach makes it inexpensive to model and animate a large library of trees for graphics applications. Figure 1 gives an illustrative summary of the process.

Neubert et al. [2007] summarize current tree modeling methods by three categories: rules-based generation, interactive modeling, and image-based production. The first group uses rule-systems such as L-systems [Lindenmayer 1968; Prusinkiewicz and Lindenmayer 1990] or procedural models [Deussen and Lintermann 2005] to generate new trees from an initial state. Talton et al. [2011] present an algorithm for high level controlling grammar-based procedural models and demonstrate the algorithm on tree modeling. Rules tend to be abstract and so are best suited to technical users, yet this is the only current group of methods capable of creating many distinct individual trees. The second group uses interaction to sketch a model in 2D and then create a 3D model from that [Anastacio et al. 2006; Quan et al. 2006; Okabe et al. 2005; Chen et al. 2008]. They provide considerable control to artists skilled enough to create high quality trees. Some methods combine rules and interaction [Lintermann and Deussen 1999; Palubicki et al. 2009]. The third group models trees from image data, with the advantage of increasing realism. Martinez et al. [2004] use a set of registered images to define a model, Neubert et al [2007] allow the construction from two loosely coupled images. Tan et al. [2007; 2008] mixes image input with user interaction to construct trees. Other approaches deal with reconstructing tree models from point clouds [Xu et al. 2007;

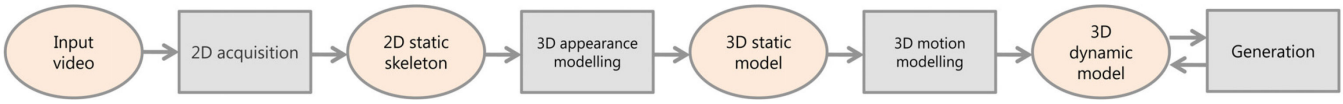


Figure 2: *Our system contains four components, indicated by four grey blocks. The first provides a 2D skeleton from the input; the second constructs a static 3D model; the third recovers motion in 3D; finally the 3D model is used as an example to generate new trees.*

Livny et al. 2010] in which case the 3D shape is implicitly given. However, expensive hardware has to be used and in most cases each tree needs to be individually edited by an expert. The latest advance in this field produces lobe-based trees [Livny et al. 2011]: the shape of the lobes is computed from the points and is a simple triangular geometry (alpha shapes). This enables storing a tree model in kilobyte and to reconstruct it in milliseconds.

All the above produce high quality static trees — making trees move has been a separate issue. Physically based approaches to motion (e.g. [Shinya and A 1992; Sakaguchi and Ohya 1999; Ota et al. 2004; Akagi and Kitajima 2006]) are computationally expensive. Heuristics have been proposed to reduce overheads [Wessélen and Seipel 2005]. Recent advances considering a tree as a harmonic oscillator [Diener et al. 2009; Habel et al. 2009] are fast enough to operate over forests. Simulations are analogous to rules for growing static trees in so far as the equations used constitute rules. Like rule-based systems, simulations can be difficult for non technical users to understand, although recent research addresses this [Diener et al. 2009]. Analogous to image based modeling, an alternative to simulation and heuristics is to use video as a data source. Diener *at al* [2006] provide an example of this. They are able to capture the dynamics of small trees in a controlled environment, and re-target the motion to large trees in the wild. However, as a deterministic approach, they rely on accurate tracking to parameterize branch motion, so the performance is less plausible when the tracking is noise. Meantime, their algorithm only uses rotations in the 2D plane to move branches so the result appears unconvincing from the side view.

We use a single video as a source to build 3D dynamic tree models, but differ from the above methods in the following aspects:

- We use a probabilistic approach to improve the appearance of 3D trees. Our method optimizes the global branch distribution whilst avoiding implausible local branching patterns.
- We recover realistic 3D motion from a single video. Again a probabilistic approach is used to account for complicated tree motion and clutter backgrounds.
- We are able to create new trees that are similar but not identical to a given example. This allows the user to create whole forests from a small set of reconstructed trees.

We use video as source because of its realism and convenience. The user provides an example video of a tree in the wild and marks the outline in the initial frame, our system then automatically outputs a 3D dynamic model. A user friendly control mechanism is provided so users can easily influence the output models — for examples, controlling the overall shape of the generated trees or turning wind up or down. Besides video, other 2D input sequences such as sketches or conventionally modeled trees can be used.

2 System

Our system contains four components (see Fig. 2). The first provides a 2D skeleton from the input; the second constructs a static 3D model from a 2D skeleton; the third recovers motion in 3D; finally the 3D model is used as an example to generate new trees.

(a) (b) (c)

Figure 3: *Motivation for our probabilistic approach that fuses the global and the local constraints:* (a) *the skeleton computed using only the global constraint. Although the skeleton fills the outline, it contains implausible local branch patterns.* (b) *the skeleton computed using only the local constraint. In this case the branch pattern appears natural, but the overall skeleton fails to minimize the empty space inside of the outline.* (c) *the result of our probabilistic approach, where the global and the local constraints are fused into a probability density function that gives the optimal solution.*

Let us firstly define some technical terms that are used throughout this paper. A *dynamic tree model* is a labeled directed graph,

$$T = (X, R, A) \tag{1}$$

meaning: a set of nodes X, a set of angular motions R, and a directed adjacency matrix A. The nodes $x_i \in X$ give the skeleton its overall shape and the adjacency matrix A gives its topological structure. For example, $a_{ji} \in A$ defines a *branch* that is directed from x_j to the x_i, so $a_{ji} = (x_j, x_i)$. This paper models trees using a binary structure and defines *bifurcation* as a basic term for describing the local branch pattern. A bifurcation comprises a vector of four elements (x_i, x_j, x_k, x_l) corresponding to local root x_j, apex x_i, and local leaves x_k, x_l. Figure 8 has some illustrations: x_j is the parent node of x_i, and x_k, x_l are two children nodes of x_i; branch a_{ji} splits into a_{ik} and a_{il} at node x_i.
To move the tree we assume each branch oscillates about its local root. For each node, we set up a local coordinate system centered at the local root $x_j(t)$ and use the world basis as rotation axes. The angular motion $r_i \in R$ is defined as a list of rotations (in angles) about the local root that sway each node x_i in time: $x_i(t+1) \mapsto (x_i(t), r_i(t))$. Throughout this paper we attach the prime symbol to variables to indicate 2D data, so $T' = (X', R', A)$ represents a 2D moving skeleton.

The first component of our system builds a 2D skeleton T' from the input video. The technique we use is based on Diener *et al.* [2006] and is briefly explained in Appendix A for completeness. Based on this 2D model we perform the 3D appearance modeling that is outlined in the next section. Motion modeling is introduced in Section 4 and Section 5 explains how to generate new models from an existing example. Results are shown in Section 6.

3 3D Appearance Modeling

This section explains how to build a 3D model from a 2D tree skeleton. This problem has been studied in the context of sketch input

Figure 4: *Comparison between different algorithms for branch placement: From left to right,* ***a****): Diener's [2006] method maps a 2D skeleton to the front and back sides of a 3D ellipsoid. The resulting 3D skeleton fails to fill a volume satisfactorily. The symmetry in the approximation is artificial, especially from the side views.* ***b****): Okabe's [2005] method can lead to bifurcations that are too sharp.* ***c****) Our result smoothly fuses global and local constraints.* ***d****) Making the branches curly and adding small twists further increases the visual richness.*

[Anastacio et al. 2006; Quan et al. 2006; Chen et al. 2008] or alternatives such as [Diener et al. 2006]. Our method owes much to Okabe et al [2005] and Tan et al [2008], the basic idea of which is to spread branches in all directions uniformly inside of an enveloping surface, Ω. Such an enveloping surface is made by surface revolution (see Okabe et al [2005]). Spreading branches inside a surface is attractively simple, but can suffer from locally implausible branch shapes, and global sub-optimal filling of the volume. Our approach as illustrated in Figure 3, addresses both of these issues simultaneously because it fuses both local and global constraints.

Our basic approach consists of two steps: a copy-operation and "pushing" the resulting structures in the right form. Initially, the reconstructed 2D skeleton ([Diener et al. 2006], see Appendix A) lies on the xy-plane, so we copy it to the yz-plane and update the adjacency matrix A to create a tree with a single trunk. More complex variants of this copying exist, such as creating copies on more than two planes like Neubert *et al* [2007] did. In all cases the result is a prototype 3D model with branches confined to distinct planes.

In a second step the model has to be adapted to create a botanically plausible structure. We do this by pushing the branches away from the initial xy- and yz-planes in a perpendicular direction. The pushing is performed following a root-to-leaf traversal.

At each step i, all the descendants of x_i (the sub-tree rooted at x_i) are pushed by the same distance. The resulting new skeleton is denoted by X_i. The key problem is how to select the optimal pushing distance: we want to fill a volume and keep tree branches in a plausible shape. Here we propose a probabilistic solution, which maximizes the posterior probability defined by the Bayes' rule

$$p(x_i|\Omega, X_{i-1}) \propto p(\Omega|X_{i-1}, x_i)p(X_{i-1}, x_i). \tag{2}$$

The posterior is factorized into two terms: the local term $p(X_{i-1}, x_i)$ keeps the branch shape plausible, while the global term $p(\Omega|X_{i-1}, x_i)$ keeps the overall volumetric shape. Notice that all points x_i are in 3D. Next we explain each term in greater detail.

3.1 Local Appearance Term

The local appearance term makes sure the local branch pattern is natural after pushing the current node x_i. It is formulated as

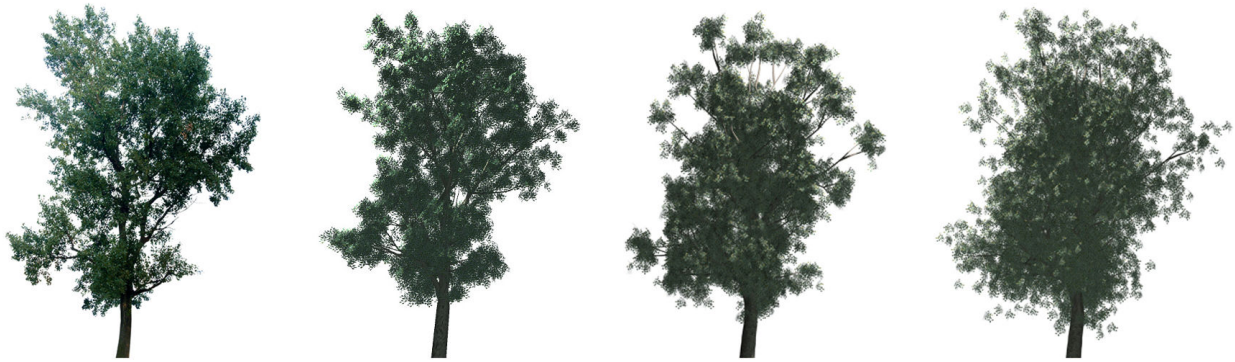
$$p(X_{i-1}, x_i) = p(X_{i-1}|x_i)p(x_i). \tag{3}$$

The prior, $p(x_i)$, is defined as a uniform distribution over a range of values $[x_i-\delta, x_i+\delta]$ along the node's pushing direction. This prior prevents a branch from being over-stretched when its tip is pushed. In practice we set δ to be $1/5$ of the width of the tree.

The conditional probability $p(X_{i-1}|x_i)$ constrains the shape of each bifurcation by examining the angle α_i between the branch a_{ji} and its parent branch

$$p(X_{i-1}|x_i) \triangleq \exp\left(-\frac{(\alpha_i-\mu)^2}{2\sigma_i^2}\right). \tag{4}$$

Here μ is the expected angle estimated as the average of all branching angles in the 2D tree. In practice we find μ lies between $\pi/6$ and $\pi/3$. Clearly $\alpha_i = \mu$ has the highest probability. σ_i controls the width of the distribution: smaller values keep the actual push closer to the expected angle μ. We set σ to depend on the topological depth of the node. Doing so allows branches close to the root



(a) (b) (c) (d)

Figure 5: *This the result of leaf density optimization.* ***a)****: The reference image.* ***b)****: Our result, where the overall shape of the leaves and its density has been optimized to match the reference image.* ***c)****: Randomly sampled leaves around branches using small variance.* ***d)****: Randomly sampled leaves using large variance.*

to have a wider tolerance: if d_i is the topological depth of the current node and d_{max} is the maximum topological depth of the tree, we set $\sigma_i = \pi(d_{max} - d_i)/d_{max}$. An intuitive explanation is that the lower branches usually form less regular bifurcations as their growths are affected by the weight of the higher order branches.

3.2 Global Appearance Term

The global appearance term keeps the overall volumetric shape. It is the conditional probability of filling the overall envelope Ω, given the previous 3D skeleton X_{i-1} updated by the current node x_i. The highest probable x_i will have the updated 3D skeleton X_i that best fits Ω. This is equivalent to minimizing the distance from Ω to X_i.

We first evenly plant m attractors, $\omega_{k=1:m}$, on the envelope surface. These attractors are used to calculate the total distance from the envelope to the 3D skeleton as

$$D(\Omega, X) = \sum_{k=1}^{m} \min(|\omega_k - X|). \tag{5}$$

The density of the attractors can be controlled by the user. We keep the total number of the attractors to be around 200 to 300. This provide good balance between the accuracy and computational efficiency. Our algorithm calculates the reduction of D when X_{i-1} is updated by x_i: $E(\Omega, X_{i-1}, x_i) = D(\Omega, X_{i-1}) - D(\Omega, X_i)$. It clamps the improvement to be non-negative, that is, $E(\Omega, X_{i-1}, x_i) = 0$ for all $D(\Omega, X_{i-1}) < D(\Omega, X_i)$. The probability is normalized over all possible solutions,

$$p(\Omega|X_{i-1}, x_i^l) = \frac{E(\Omega, X_{k-1}, x_i^l)}{\sum_{j=1}^{L} E(\Omega, X_{i-1}, x_i^l)}, \tag{6}$$

Here x_i^l is one of the L possible solutions for x_i. The probability of x_i being outside of Ω is clamped to zero, which ensures all pushed branches are inside the envelope.

Substituting Eqs. (3), (4), (6) into Eq. (2) gives the final posterior. The pushing distance that maximizes this posterior is chosen as the optimal solution for x_i. Intuitively, our system exhaustively tries out all possible "push" values within $[x_i - \delta, x_i + \delta]$, evaluating each using the quality measure that is dictated by the probability model: for each push position, the quality of the branch angle is measured using Eq. (4), and the equality of the space fill is measured using Eq. (6). Then the best push distance is selected. Figure 4 shows our output 3D skeleton and results of alternative algorithms. Notice that at each step we search for the best value to push the current node and all its direct and indirect children. So in total there are N times L possible options to convert a whole tree into 3D. Here L is the number of options for pushing a single node and N is the total number of nodes in the tree.

Finally, we follow existing methods to add details at low cost. Small branches and twigs are added using self-similarity [Shlyakhter et al. 2001; Tan et al. 2007; Chen et al. 2008] while curliness and twists are generated by fitting a Flash and Hogan parametric line model [Flash and Hogan 1984]. We also propose a simple but efficient algorithm that generates leaves that match the reference image, which is explained in Appendix B.1. The result is demonstrated in Figure 5.

4 3D Motion Modeling

So far we have modeled a static 3D tree from the given 2D input. The next step is to model its motion also in 3D. We do so using tracking data from the video. Again the key problem is to transform the tracked 2D motion to 3D that moves the model in a realistic way. Let the 2D projection of each 3D node x_i at frame t be denoted by $x_i'(t)$[1]. Video based tracking moves this point to $x_i'(t+1)$ in the next frame.

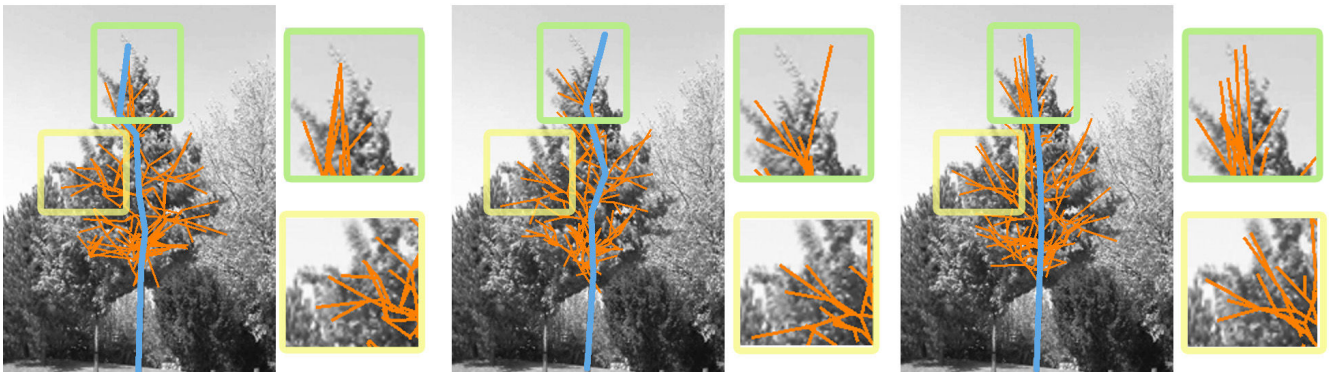
We first assume the length of the 3D branch does not change, so that any apparent 2D translation $x_i'(t+1) - x_i'(t)$ can be deterministically explained by a sequence of 3D rotations about the local root $x_j(t)$. Section 4.1 explains this approach in greater detail. In practice though, a deterministic approach leads to a sub-optimal solution, for many reasons such as errors in the tracking data. As a result the tree can be bent and twisted out of shape. Figure 6 and the supplementary material show some examples. To solve this problem we again use a probabilistic approach to recovering 3D motion, as explained in Section 4.2.

4.1 Deterministic motion estimation

Using the principle that the moving 3D tree must at all times project onto a moving 2D skeleton, we can obtain a deterministic solution for the 3D angular motion R in Eq. (1). We assume that a branch tip $x_i(t)$ can only rotate about its local root $x_j(t)$. This follows direct experience that wood can bend but is almost impossible to stretch or compress and is a consequence of the way in which fibres are made up; the Young's modulus for wood is around 10^9 to $10^{10} N/m^2$ which is comparable with many metals.[2] If a 2D branch appears

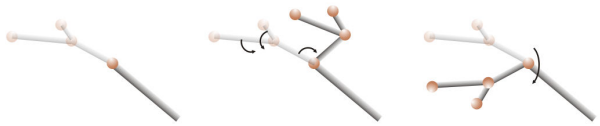
[1] please recall the prime symbol in this paper indicates 2D data

[2] http://en.wikipedia.org/wiki/Young's_modulus



(a) 3D deterministic result (b) Diener's [2006] result (c) Our result

Figure 7: *3D motion acquired from an input video. (a): a 3D deterministic method can not handle the inaccuracy in tracking (highlighted in green and yellow). So branches can be bent and twisted out of shape (in blue). (b): Diener's method also produces implausible branch shapes (in blue). Meantime it can not model branch foreshortening and produces motion that diverts from the real tree movement. (highlighted in green and yellow). (c): Our method produces realistic 3D motion while keeping its projection following the video.*



(a) (b) (c)

Figure 6: *Motivation for the proposed probabilistic 3D motion modeling. This figure demonstrates examples of angular motion that lead to implausible branch shape.* ***a)****: The original branches.* ***b)****: The branch is bent out of shape.* ***c)****: The branch is over twisted when its parent aligns with a rotation axes. Both problems are solved use our approach.*

to change length, it can only be because the 3D branch has been foreshortened by projection. The idea here is to rotate the branch to allow for this foreshortening.

We set up a local coordinate system centered at the local root $x_j(t)$ and use the world basis as rotation axes. Rotating $x_i(t)$ around its local root gives Φ, the sphere of all possible positions in the next frame t. Using [Lucas and Kanade 1981], we also have the node's 2D projection tracked in the next frame, namely $x_i'(t+1)$. The 3D position $x_i(t+1)$ must lie on the intersection of Φ and the line of sight (aligned with the z direction) through $x_i'(t+1)$. In practice, tracking data is noisy so there can be two, one or zero intersections, depending on the distance between $x_i'(t)$ and $x_i'(t+1)'$ and the size of Φ. In the case of two intersections, we select the solution with the closer 3D distance to $x_i(t)$. In the case of no intersection, we shift $x_i'(t+1)$ towards $x_i'(t)$ until a solution is found.

This simple deterministic approach usually complies with the motion in the original video. However, the appearance of the resulting skeleton is not always optimal. As seen in Figure 6 and 7 the branch can be bent out of shape or twist unrealistically when the tree moves with large motion. This is because deterministic approaches rely on near-perfect tracking data and do not regularize the result for natural branch shapes. Such a problem is shared by Diener *at al* [2006], who also use a deterministic approach but only model the motion as 2D rotations in the xy plane. It is worth mentioning that 2D plane rotations are insufficient for modeling many 3D movement, such as the foreshortening effect from the front view (see Figure 7 middle) and so the resulting motion appears unconvincing from the side view (see the supplementary video). Such artifacts of the deterministic approaches motivate our use of probabilistic methods.

4.2 Probabilistic Motion Modeling

As with the 3D appearance modeling, our motion modeling also follows a root to leaf traversal for each frame. Also, any motion applied to a branch node $x_i(t)$ is applied to all of its descendants. Recall the problem is, given a 3D skeleton and the track of its 2D projection, to find the optimal 3D angular motion, $r_i(t)$, for each node $x_i(t)$. Using the Bayes's rule, the probability of choosing a particular $r_i(t)$ can be factorized into two terms,

$$p(r_i(t)|x_i'(t+1), X_{i-1}(t+1)) \propto p(x_i'(t+1)|X_{i-1}(t+1), r_i(t))p(X_{i-1}(t+1), r_i(t)). \quad (7)$$

We use $X_{i-1}(t+1)$ to indicate the skeleton before rotating $x_i(t)$, where all ancestors of $x_i(t)$ have been rotated to the next frame $t+1$. The "shape" term $p(X_{i-1}(t+1), r_i(t))$ evaluates whether the skeleton has natural local branch patterns, after $x_i(t)$ is rotated by $r_i(t)$. It is similar to the previously discussed local appearance term (Section 3.1). The "track" term $p(x_i'(t+1)|X_{i-1}(t+1), r_i(t))$ evaluates the similarity between the (projected) rotated branches and their 2D tracked motion; in other words, how well the rotation matches the track. It is analogous to the global appearance term (Section 3.2), which evaluates the fit between the skeleton and the envelop surface. As before, primes indicate 2D data.

4.2.1 Track Motion Term

The track motion term $p(x_i'(t+1)|X_{i-1}(t+1), r_i(t))$ is the probability that the rotation matches the 2D tracked motion $x_i'(t+1)$. The node $x_i(t)$ undergoes a putative 3D rotation around its local root, and is projected to 2D under a constant orthogonal projection matrix K: that is, $y_i'(t+1) = K(x_i(t), r_i(t))$. We define the probability of the observed tracking using the distance between the predicted point $y_i'(t+1)$ and the observed point $x_i'(t+1)$:

$$p(x_i'(t+1)|X_{i-1}(t+1), r_i(t)) = \exp\left\{\frac{(y_i'(t+1) - x_i'(t+1))^2}{-2\sigma_{track}^2}\right\}. \quad (8)$$

Since the rotation term is the only variable, the rotation leading to the predicted point $y_i'(t+1)$ with highest probability will be favored; σ_{track} is a pre-set parameter that controls the precision of this estimate. If this were the only expression used to decide the rotation we would recover the deterministic solution once again. Next we introduce the shape motion term as a prior which prevents the branch from bending out of shape.

Figure 8: *Trees generated by statistical sampling. The first tree in each row is the example model, that used to generate new trees on the right. The top row and the bottom row use different Dirichlet distributions to generate different bifurcation shapes. As an example, the Dirichlet distributions of ratio of angles between the branches are illustrated on the left.*

4.2.2 Shape Motion Term

The shape motion term constrains the rotated skeleton to a natural appearance. It can be further decomposed into two terms,

$$p(X_{i-1}(t+1), r_i(t)) = p(X_{i-1}(t+1)|r_i(t))p(r_i(t)). \qquad (9)$$

Here, $p(r_i(t))$ is the prior of the angular motion, that is a rotation we might guess at if we have no track data at all. We define this prior as a uniform distribution over $[-\delta, \delta]$, so the instant angular velocity for each node is bounded.

A threshold Δ is also set for the accumulated angular motion over all the past frames, so that $|\sum_{\tau=0}^{t} r_i(t)| > \Delta$ is assigned with zero probability. This is an important constraint when the branch aligns with a rotating axis. Without it, an arbitrary rotation around that axis would satisfy Eq. (9), even though the skeleton is twisted unnaturally as time moves on (see Figure 6*c*).

The conditional probability $p(X_{i-1}(t+1)|r_i(t))$ evaluates the naturalness of the shape of the skeleton after rotating node $x_i(t)$ by $r_i(t)$. It is controlled by the angle $\alpha_{ij}(t)$ between the current branch and its parent branch. We set

$$p(X_{i-1}(t)|r_i(t)) = \exp\Big\{-\frac{(\alpha_{ji}(t) - \alpha_{ji}(1))^2}{2\sigma^2_{shape}}\Big\}. \qquad (10)$$

Here the $\alpha_{ij}(1)$ is the angle between branch a_{ji} and its parent branch in the initial frame. σ_{shape} is a free parameter that can be set to vary with the topological depth, similar to σ_i in Eq. (4). For example, a wider tolerance σ_{shape} can be assigned to nodes near the leaves so they bend more easily.

The final posterior is calculated by substituting Eqs. (8), (9), (10) into Eq. (7). The angular motion that maximizes this posterior is used to rotate x_i. To demonstrate the advantage of our probabilistic based method, Figure 7 compares 3D branch animations using the motion acquired from the deterministic approaches and our method. As the figure shows, the deterministic method can bend branches out of shape, while our method preserves its realism.

So far we have modeled the motion of a bare tree, the next step is to generate motion for the leaves. The basic leaf motion is to follow the branch it is attached to, with high frequency movement introduced to enrich the dynamic. See Appendix B.2 for detail.

5 Generation

So far we have modeled a single tree. In this section, we introduce a method that generates multiple unique trees that look and move similarly, but not identically, to an existing model. The existing model can be the one we just modeled from a video. However, this module makes no assumption about the source of the example, so a manually created model would be equally acceptable. The advantage of having this module is that a potentially infinite number of trees can be automatically created. Rule based systems can do this too, but our method relieves the user of the need to define rules. Instead our system automatically learns a statistical model of an input example, and uses that statistical model to make new trees. Hence it provides a unique solution to the tree modeling problem.

One way to generate new trees from an existing one is to add some randomness to the example. For example, jitter branches by adding Gaussian noise. However, keeping the randomly jittered trees looking natural is not a negligible task. Moreover, only limited novelty can be produced — for example, the global branch distribution re-

mains similar. The same problems exist for motion too.

We propose to make use of a statistical model to generate new trees. The new trees and the example tree may look and move differently, but they should share a common distribution of characteristic features and so are statistically identical. Figure 8 shows some example trees automatically generated from examples. Broadly speaking, our algorithm grows a completely new model from root to leaves, adding a bifurcation at each recursive step. The shape of the bifurcation depends on tree type and so follows the statistics in the example tree. A new tree is first generated in its static form in 2D, then converted into 3D tree, using the 3D modeling method explained in Section 3. Motion characteristics are then included to allow the tree to sway naturally over time. The "stiffness" of the tree is reflected in the frequencies that dominate the swaying motion of its branches. The statistics of these frequencies in the example tree are used to generate a statistically identical motion in the example.

The key ingredient of the above method is to learn the statistical characters of the example tree. In order to do this, we first define features that characterize the appearance and motion of a tree. The features are represented as a points in a feature space. We then fit a parametric distribution to the collection of feature points taken from the example. New features can be sampled from the distribution for the new tree. Thus we have a probabilistic generative model (PGM). Figure 8 shows different types of tree have different distributions — that is, different PGMs. Next we first discuss the strategy for feature selection and PGM fitting in detail, then explain the algorithm to grow a new tree by assembling these features.

5.1 Feature Selection

The United States Forestry Service states that leaf shape and color, and the tree's general shape are all useful indicators of a tree's class (USFS-TAMU). Our method uses the shape of a tree as a 2D envelope surface to constraint the overall shape of the generated tree and the shape of bifurcations to characterize a tree species. As mentioned, we also use dominant swaying frequencies to characterize motion. The envelope surface is used when growing the tree, here we focus on the shape and motion of bifurcations.

Recall that in a binary tree all bifurcations have exactly four elements: the local root x_j, the apex x_i, and the local leaves x_k, x_l(see Figure 8 for an illustration). We can describe the shape of a bifurcation with a feature vector of fixed dimension. The feature vector for the shape of a bifurcation should not depend on the location, orientation or the absolute scale of the bifurcation — otherwise viewing the same tree from different positions would give different results. The features we choose comprise: (i) the ratio of angle between the branches, $< \theta_1 : \theta_2 : \theta_3 >$ so $\sum_i \theta = 1$; and (ii) the ratio of branch lengths $< L_1 : L_2 : L_3 >$ so that $\sum L_i = 1$. The summation constraint turns out to be crucial in the later PGM fitting.

For motion, we assume a branch, a_{ji}, will sway according to dominant frequencies. The ratio between the energy of each given frequency also sum to unity. This makes it possible for us to represent motion using the same statistical model as we use for shapes, but in higher dimensions. More exactly, we keep the lower frequency components (the first 20% or so) as the dominant frequencies. Suppose that $c_k + id_k$ is a complex number representing the real and the imaginary parts of the k^{th} frequency component. We represent dominant frequencies with a pair of tuples, $< c_k >_1^n$ and $< d_k >_1^n$, both of which are scaled to sum to unity. For the purpose of fitting a PGM, the tuples are shifted so that no term is negative. This shift is removed after sampling from the fitted PGM.

5.2 PGM Fitting and Sampling

We compute shape and motion features for every node in the example tree. When plotted in feature space, these features form a tight cloud. We fit a parametric function whose contours follow paths of equal density through this cloud. Notice that the summation constraint means the feature vectors are really tuples of ratios. The important consequence of this is that a Gaussian distribution is inappropriate to model the distribution — because it cannot guarantee the summation constraint. Instead we use the Dirichlet distribution, which is designed for tuples with exactly the summation constraint we have. The Dirichlet distribution is briefly explained next, but see [Bishop 2006] for a more in-depth exposition.

Let $\mathbf{z}$ denotes a random vector whose elements satisfy $z_k > 0$ and $\sum z_k = 1$. Here z_k represents the proportion of item k, i.e. the proportion of the kth bifurcation angle in all the three angles. The probability density function of this random vector $\mathbf{z}$ is then modeled by a Dirichlet distribution with the parameter vector $\boldsymbol{\beta}$:

$$p(\mathbf{z}|\boldsymbol{\beta}) \sim D(z_1, ..., z_K|\beta_1, ..., \beta_K) = \frac{\Gamma(\Sigma_k \beta_k)}{\prod_k^K \Gamma(\beta_k)} \prod_k^K z_k^{\beta_k - 1}, \quad (11)$$

where Γ denotes a gamma function. $\boldsymbol{\beta}$ can be learned from some training data, i.e. an existing tree model (see [Minka 2003] for details). Notice the length of the parameter vector $\boldsymbol{\beta}$ is the same as the random vector $\mathbf{z}$. $\boldsymbol{\beta}$ can be understood by its precision $s = \sum_k \beta^k$ and the mean of the distribution $\mathbf{m} = \frac{\boldsymbol{\beta}}{s}$. Intuitively, beta controls the shape of the distribution: $\mathbf{z}$ is likely to be near the mean when s is large, and distributed more diffusely when s is small.

We model different tree features independently because they spread into different feature dimensions and shapes. We define $\boldsymbol{\beta}_{angle}$, $\boldsymbol{\beta}_{length}$ and $\boldsymbol{\beta}_{motion}$ as the parameter vectors for the branch angle, length and motion. To model bifurcation shapes, $\boldsymbol{\beta}_{angle}$ and $\boldsymbol{\beta}_{length}$ only need to be 3D vectors. It is slightly more complicated for $\boldsymbol{\beta}_{motion}$: for each rotating axes, we have its own $\boldsymbol{\beta}$, which is learned in the frequency domain rather than in the spatial domain: we perform the fourier transform to the time series of the branch angular motion and only keep the most informative (lowest 20%) frequencies.

5.3 Growing a New Tree

A new tree is first generated in 2D, then converted into 3D. The Dirichlet distribution ensures each bifurcation has a well formed shape and moves naturally, but does not guarantee anything about the global shape of the new tree. To ensure this we need a space-filling constraint, which is analogous to the problem we faced when converting a 2D skeleton to 3D. As in that case we need a 3D envelop surface, on this occasion denoted Ω' to indicate its 2D nature.

We grow a 2D tree from root to leaf to fill Ω'. At each step a leaf point is randomly chosen and a new bifurcation is added to it: the bifurcation is sampled from the PGM. The growing direction of the new bifurcation is decided by two criteria: (i) it should efficiently reduce the empty space in Ω', and (ii) the angle between the newly added bifurcation and its parent should appear natural. These two criteria are equivalent to those used when converting a skeleton to 3D, so we re-use the approach except: (a) all spatial terms are now in 2D, (b) rather than "push" we find the best growing direction. This is equivalent to finding the best rotation of the child branch around the z axis. This is essentially a one dimensional search problem and the optimal solution can be efficiently found.

Leaves are added to a generated tree using standard approaches [Tan et al. 2008]; if greater control is required for a specific tree the user can provide a hand-drawn density map. Figure 8 shows

some results of trees generated into different types. The Dirichlet distributions in the figure are visualizations of bifurcation angle triplets, similar distributions exist for branch lengths triplets and sway frequency in higher dimensions. The output tree looks and moves like the input example (see supplementary video for motion results), but is not identical. Generation is unique and important to the current tree modeling literature because it makes populating large landscape scenes straightforward for users.

6 Results

Figures 1 and 10 show our method works for different tree species. Each row shows the results from a single input video. The left column shows the initial frames from the videos, with backgrounds removed. The middle column shows our output model from the front view. Note that our output model matches the reference image from the front and has a natural appearance from all the other angles. The right column shows new trees generated with similar appearances. The ability to generate unique trees of the same species makes populating large landscape scenes straightforward. In the supplementary video we show examples of moving trees, that demonstrate the robust performance of our method for complicated tree movement and cluttered background. In comparison, the deterministic methods fail to realistically capture the motion once the tracking data appears noisy.

Our system uses video as input because it is convenient for users and offers realism that is expensive to achieve otherwise. However, there is no reason why the system could not work with other types of input source - for example, sketch based input as shown in Figure 9. Indeed, sketch based input can be used to constrain the branching structure for a conifer tree. In practice, the central trunk is initialized by the user and will not be copied nor pushed. In general, our system can model trees that have a skeletal structure which can be represented by a graph of nodes and arcs.

Our system provides flexible user control for editing the output models. For example, the appearance of the tree can be controlled by a 2D outline supplied by a user. Typically this outline can be recycled from the user outlining a tree in a video frame, but the opportunity for creative control exists allowing users to indulge in topiary, for example. Figure 11 shows how 3D tree branches optimally fill unusually shaped volumes, and leaves can still be added to complete the model. As leaves add considerable complexity to a model in terms of the number of polygons, we allow users to control the complexity of the model by balancing the number of seeds and the scale of the template. To make a lower resolution model, users only need to sub-sample the seeds and scale the template to a larger size. See supplementary video for an example. Our model stores the frequencies of branch oscillations at each node, which means users can exercise control over how the tree moves through a driving signal that effects the oscillations. For example, the user can drive a tree using an audio signal of a windy day. See the supplementary video for an example of motion magnification.

The major free parameters are generally fixed for different trees mentioned in the paper. For example, the search range δ for $p(x_i)$ in Eq.(3) is fixed to be 1/5 of the width of the tree. The standard deviation σ_i in Eq.(4) is calculated from the topology depth of the branch. However, a few parameters may be adjusted slightly for specific scenarios. For example, the bound of the angular velocity for $p(r_i(t))$ in Eq.(9) is enlarged to cope with the strong wind example in the demo video. In practice, the time taken to build a static tree is within a minute using our combined Matlab and C++ implementation running on a standard Intel P4 desktop. 3D motion generation usually takes about one extra second per frame. The memory consumption is small, the system consumes about 30 Mb memory while running. The output model can be between 10K and 100K polygons, depending on the required resolution.

A user specified 2D sketch. Our result.

Figure 9: *User sketch helps model some specific type of trees. The main central of the conifer tree (in orange) is specified to not being copied nor pushed in the 3D modeling step. User can also draw further branch details (in gray) to increase the similarity between the model and the input example.*

Ideally results should be quantitatively evaluated. In practice we realize it is difficult to get the ground truth data for trees. Recent advances such as [Livny et al. 2010] uses a scanning technique to acquire 3D points clouds for trees. Although using an active scanner is a step forward toward highly accurate geometry capturing for trees, the data is still sparse, incomplete, and noisy so can not be used as the ground truth. Here we would like to emphasize that the purpose of our work is to provide a single view solution for a visually plausible model. And we have provided qualitative examples, including comparisons with other results, comparisons with the reference video and a number of graphics applications.

For a single tree modeling, our work draws inspirations from existing art, such as [Okabe et al. 2005; Tan et al. 2007; Chen et al. 2008]. Particularly, [Chen et al. 2008] proposed a Markov random field approach to convert a freehand sketch of a tree into a full 3D model that is both complex and realistic-looking. Like [Chen et al. 2008], our approach pushes a 2D skeleton model into 3D from root to leaf, but whereas [Chen et al. 2008] grow branches in random directions we supply a constraint (that factors into local and global parts). In addition, our Bayesian framework is also used to model 3D motion, whereas [Chen et al. 2008] only works with static models.

Last but not least, we discuss the range of tree species that our method is designed to work with. According to the list of tree genera[3], the major tree species include Eudicots, Monocotyledons, Magnoliids, Conifers, Ginkgos, Cycads, and Ferns. As a skeleton based approach, our method works better with trees that have an adequate branching structure. These include Eudicots, Magnoliids, Conifers and Ginkgos. Although not all of these trees strictly follow a binary skeleton, we found our method produces visually plausible output models. In addition, it is very convenient for users to add further constraints to improve the realism of a particular tree species. For example, the central trunk of conifer trees typically does not branch like a deciduous tree. In this case the user can enforce this at the initialization stage (Fig. 9). However, our system works less well on Monocotyledons, Cycads and Ferns, which are leaf-dominated

[3] http://en.wikipedia.org/wiki/List_of_tree_genera

species. For example, a palm tree is expected to appear as a solitary shoot ending in a crown of leaves. In this case, specific prior knowledge about the tree species is needed to acquire a plausible output model.

7 Conclusions

In this paper we described a method for reconstructing animated 3D tree models from 2D video input and showed how to create a variety of models automatically from a reconstructed tree. Our probabilistic formulation fuses local and global constraints to ensure optimal static models. A probabilistic method is also used in motion modeling to fuse the shape and track terms, which maintains the integrity of branch shapes in 3D dynamic models. Our approach to using video reduces user interaction to outlining a tree in just one frame. The user can control the shape of generated trees by drawing a new outline, and control the complexity 3D models by specifying the number of leaves. Furthermore, the motion can also be controlled by modulating the dominant frequencies stored at each node.

We have provided side-by-side comparisons to other work to show the qualitative advantage of our models. Indeed, we have not provided quantitative evaluation. However, our system is designed for graphics application, where visual satisfaction is highly valued and perfect reconstruction is less demanding.

Our system needs a 2D skeleton model to work with. The 2D model can be acquired from a single input video for the realism it offers. Our system is also compatible with other forms of input data, including user sketches, as long as the skeleton model can be expressed as a graph of nodes and arcs. However, palm trees is a limiting case as they are not characterized by branches but leaves.

A more important restriction is the use of binary skeletons. So far we have not found this to be significant in practice, and is it very convenient for automatic generation: allowing a general n-ary tree introduces a significant degree of complexity into the generative model. However, this can be an interesting path for future work.

Another area ripe for further development lies in the simultaneous modeling of motion and shape in 3D. Instead of first modeling a static 3D skeleton, then moving it we might consider solving for the moving skeleton that best fits all frames simultaneously. Using multiple views should further improve the realism of our output models. Last but not the least, it is interesting to explore whether our data driven approach can be used to parameterize physical models such as used in [Diener et al. 2009; Habel et al. 2009; Sun et al. 2003], so the data can be better applied in realtime graphics applications.

References

AKAGI, Y., AND KITAJIMA, K. 2006. Computer animations of swaying trees based on physical animation. *Computers and Graphics 30*, 4, 529–539.

ANASTACIO, F., SOUSA, M. C., SAMAVATI, F., AND JORGE, J. A. 2006. Modeling plant structures using concept sketches. *Proceedings of the 4th international symposium on Non-photorealistic animation and rendering*, 105–113.

BISHOP, C. 2006. *Pattern Recognition and Machine Learning*. Springer-Velrag.

CHEN, X., NEUBERT, B., XU, Y.-Q., DEUSSEN, O., AND KANG, S. B. 2008. Sketch-based tree modeling using markov random field. *ACM Trans. Graph. 27*, 5, 1–9.

DEUSSEN, O., AND LINTERMANN, B. 2005. *Digital Design of Nature: Computer Generated Plants and Organics*. Springer-Verlag.

DIENER, J., REVERET, L., AND FIUME, E. 2006. Hierarchical retargetting of 2d motion fields to the animation of 3d plant models. *ACM SIGGRAPH/Eurographics Symposium on Computer animation*, 187–195.

DIENER, J., RODRIGUEZ, M., BABOUD, L., AND REVERET, L. 2009. Wind projection basis for real-time animation of trees. *Computer Graphics Forum (Proceedings Eurographics 2009) 28*, 2, 533–540.

FLASH, T., AND HOGAN, N. 1984. The coordination of arm movements: An experimentally confirmed mathematical model. *Journal of Neuroscience 5*, 1688–1703.

HABEL, R., KUSTERNIG, A., AND WIMMER, M. 2009. Physically guided animation of trees. *Computer Graphics Forum (Proceedings Eurographics 2009) 28*, 2, 523–532.

HARRIS, C., AND STEPHENS, M. 1988. In *Proc. 4th Alvey Vision Conference*, 189–192.

LINDENMAYER, A. 1968. Mathematical models for cellular interactions in development ii. simple and branching filaments with two-sided inputs. *Journal of Theoretical Biology 18*, 3, 300–315.

LINTERMANN, B., AND DEUSSEN, O. 1999. Interactive modeling of plants. *IEEE Computer Graphics and Applications 19*, 1, 56–65.

LIU, C., TORRALBA, A., FREEMAN, W. T., DURAND, F., AND ADELSON, E. H. 2005. Motion magnification. In *ACM SIGGRAPH*, 519–526.

LIVNY, Y., YAN, F., OLSON, M., CHEN, B., ZHANG, H., AND EL-SANA, J. 2010. Automatic reconstruction of tree skeletal structures from point clouds. *ACM Trans. Graph. 29* (December), 151:1–151:8.

LIVNY, Y., PIRK, S., CHENG, Z., YAN, F., DEUSSEN, O., COHEN-OR, D., AND CHEN, B. 2011. Texture-lobes for tree modeling. *ACM Siggraph*, to appear.

LUCAS, B. D., AND KANADE, T. 1981. An iterative image registration technique with an application to stereo vision. *Proceedings of the 7th International Joint Conference on Artificial Intelligence*, 674–679.

MINKA, T. P. 2003. Estimating a dirichlet distribution. *M.I.T Technical report*.

NEUBERT, B., FRANKEN, T., AND DEUSSEN, O. 2007. Approximate image-based tree-modeling using particle flows. *ACM Trans. Graph. 26*, 3, 88–95.

OKABE, M., OWADA, S., AND IGARASHI, T. 2005. Interactive design of botanical trees using freehand sketches and example-based editing. *Comput. Graph. Forum 24*, 3, 487–496.

OTA, S., TAMURA, M., FUJIMOTO, T., AND K, M. 2004. A hybrid method for the real-time animation of trees swaying in wind fields. *The Visual Computer 20*, 11, 613–623.

PALUBICKI, W., HOREL, K., LONGAY, S., RUNIONS, A., LANE, B., MĚCH, R., AND PRUSINKIEWICZ, P. 2009. Self-organizing tree models for image synthesis. *ACM SIGGRAPH*, 1–10.

PRUSINKIEWICZ, P., AND LINDENMAYER, A. 1990. *The algorithmic beauty of plants*. Springer-Verlag.

QUAN, L., TAN, P., ZENG, G., YUAN, L., WANG, J., AND KANG, S. B. 2006. Image-based plant modeling. *ACM Trans. Graph. 25*, 3, 599–604.

RECHE-MARTINEZ, A., MARTIN, I., AND DRETTAKIS, G. 2004. Volumetric reconstruction and interactive rendering of trees from photographs. *ACM Trans. Graph. 23*, 3, 720–727.

SAKAGUCHI, T., AND OHYA, J. 1999. Modeling and animation of botanical trees for interactive virtual environments. In *ACM symposium on Virtual reality software and technology*, 139 – 146.

SHI, J., AND MALIK, J. 2000. Normalized cuts and image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell. 22*, 8, 888–905.

SHINYA, M., AND A, F. 1992. Stochastic motion-motion under the influence of wind. *Computer Graphics Forum 11*, 3, 119–128.

SHLYAKHTER, I., ROZENOER, M., DORSEY, J., AND TELLER, S. 2001. Reconstructing 3d tree models from instrumented photographs. *IEEE Comput. Graph. Appl. 21*, 3, 53–61.

SUN, M., JEPSON., A. D., AND FIUME, E. 2003. Video input driven animation (vida). In *Proceedings of the Ninth IEEE International Conference on Computer Vision - Volume 2*, 96–103.

TALTON, J. O., LOU, Y., LESSER, S., DUKE, J., MĚCH, R., AND KOLTUN, V. 2011. Metropolis procedural modeling. *ACM Trans. Graph. 30*, 11:1–11:14.

TAN, P., ZENG, G., WANG, J., KANG, S. B., AND QUAN, L. 2007. Image-based tree modeling. In *ACM SIGGRAPH*, 87 – 93.

TAN, P., FANG, T., XIAO, J., ZHAO, P., AND QUAN, L. 2008. Single image tree modeling. *ACM Trans. Graph. 27*, 5, 1–7.

WESSÉLEN, D., AND SEIPEL, S. 2005. Real-time visualisation of animated trees. *The Visual Computer 21*, 6, 397–405.

XU, H., GOSSETT, N., AND CHEN, B. 2007. Knowledge and heuristic-based modeling of laser-scanned trees. *ACM Trans. Gr. 26*, 4, 19–31.

A 2D Skeleton Acquisition

We use an approach based on Diener *et al.* [2009] to acquire a 2D skeleton from a single input video. The process is briefly explained here for completeness.

First the user needs to outline the tree in frame one, because automatic segmentation is not solved. Given an outline we detect Harris interest points [1988] on the leafy part of the tree. The Harris points are tracked from frame to frame. Our system then constructs a hierarchy using recursive binary clustering, first dividing the whole feature set into halves, then each half into quarters and so on. We use the *Normalized Cut* algorithm [Shi and Malik 2000] for clustering because it is designed to partition data based on affinities between pairs on nodes. An affinity matrix is calculated from the spatial distances between the features and their similarity of motion [Liu et al. 2005]. The clustering process stops when the hierarchy reaches a certain level. We find $5-7$ levels are sufficient for many trees species without introducing unnecessary complexity. The output is a binary hierarchy, and the links between the parent and the children nodes are stored in the adjacency matrix A.

The hierarchy can be used for tracking the tree over long periods of time, which implies it is a plausible model of the branching structure. However, the hierarchy is not well suited for graphics purposes because it does not look like a physical skeleton of branches. To find a skeleton we use the centroid of each cluster as an initial approximation of nodes in X'. Next we shift $x' \in X'$ to a better position: for every branch a_{ji}, we shift the child node along a line x'_i towards its parent: $x'_i \mapsto x'_i + \alpha(x'_j - x'_i)$. The process is recursive, from root to tip nodes. The effect is to "fold" the initial skeleton to a visually acceptable form, and one which tracks the video. In our experience, $\alpha = 0.4$ suits many tree types. The folding process stops at the second last level to prevent an overall shrinking of the skeleton.

B Leaf Modeling

B.1 Leaf Appearance Modeling

Leaves can be generated using random sampling [Tan et al. 2008]. However, randomly sampled leaves tend to be unsatisfactory when compared with the real reference image. The problem is shown in Figure 5(c-d), where the overall leaf density does not match the real tree in Figure 5(a). Here we propose a simple but efficient algorithm that outputs the optimal leaf density, as shown in Figure 5(b).

We use Harris interest points detected in section A as 2D "seeds" to generate 3D leaves. Seeds are divided into two halves based on their intensities. Each seed is then assigned to a branch based on their Euclidian distances in the xy plane. Dark seeds are assigned to branches at the back of the tree; bright seeds are assigned to branches at the front. The bright and dark leaf division improves the realism of the shading process.

Now we have seeds in 3D, the next step is to generate leaf details such as shape, orientation and texture. For each tree, we have a user-defined 3D mesh model that represents a cluster of leaves. We locate one mesh model at each seed. The orientation of the mesh is adjusted according to its location so the leaves naturally face different directions around the tree. For photo-realistic rendering, user-defined texture is mapped to the mesh.

B.2 Leaf Motion Modeling

[Habel et al. 2009] proposed a physically guided approach to animate leaves. Different from their method, we approximate the basic leaf motion to follow the branch it is attached to. However, this usually results in motion that looks damped. On the other hand, independently jittering the motion of each leaf looks wrong too, because the motion of leaves are tied together by the twigs they grow from.

To solve this problem we introduce some random motion to rotate leaf clusters around their main axis. In addition to the underlying branch motion, we oscillate each leaf cluster using a sine wave: $y(t) = Asin(\omega t + \beta)$. β is calculated as the distance from the center of each leaf cluster to the branch node it is assigned to, so each cluster starts with a slightly different phase. The amplitude A is modulated by the magnitude of the branch velocity, so larger branch motion results in larger difference between the leaves. ω is a user tunable parameter that controls the frequency of the sine wave. Now the motion of each leaf can be enriched in a similar way: for each polygon, we select a vertex as the apex for rotation and introduce high frequency angular motion to it.

Acknowledgements

We thank reviewers for their valuable suggestions. We would also like to thank UK's Engineering and Physical Sciences Research Council for supporting this work with grant EP/D064155/1, and for supporting the Media Technology Research Centre and the Centre for Digital Entertainment at University of Bath.

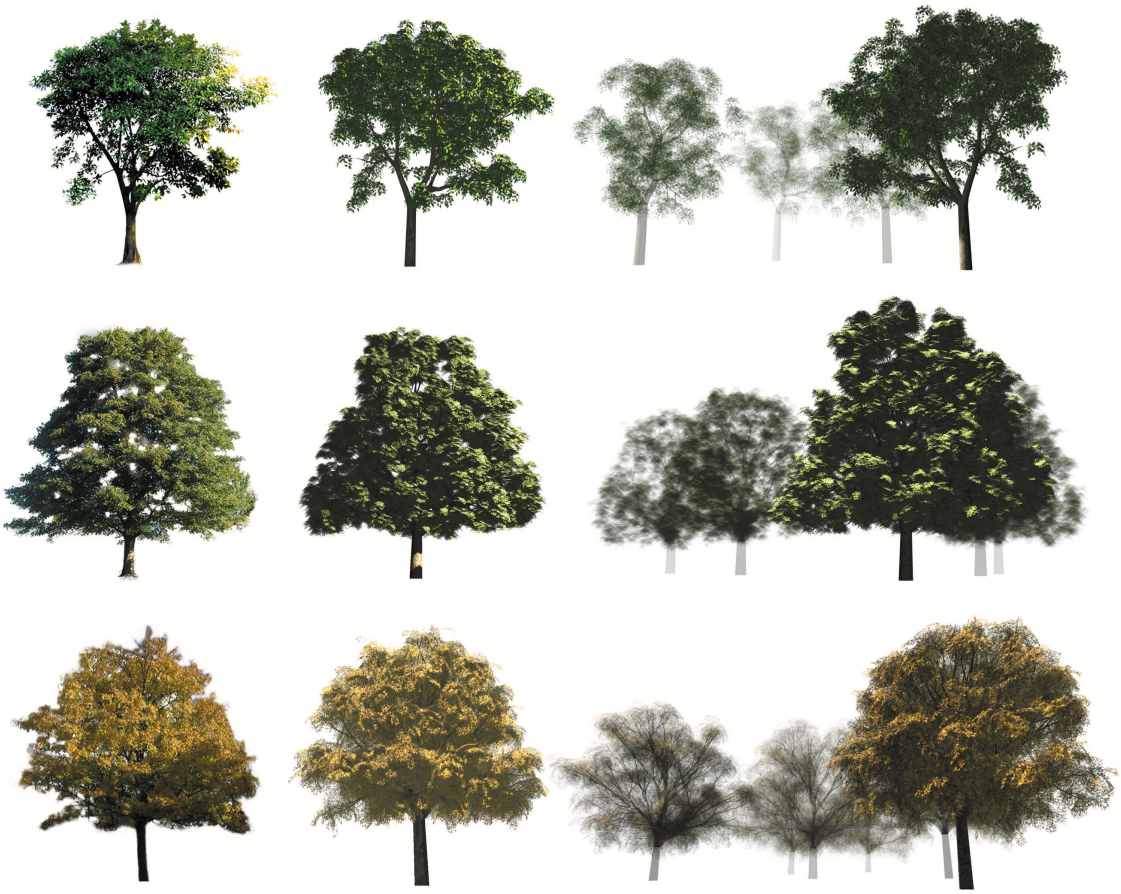


Figure 10: *Generation of different types of tree. Left: an initial frame of the input video. Background is removed using alpha matting. Middle: the output 3D tree model. Right: some generated new models, using the middle one as the example.*

Figure 11: *Tree topiary: users can control the shape of the generated trees using different envelope shapes. Top row: the resulting trees using a triangle, a circle, a pentagon and a heart shape as the envelopes. In the bottom row a vase image is used to extract an outline.*